UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,190	11/29/2003	Douglas Wooff	50325-0842	9853

29989 7590 08/29/2008

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
08/29/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/725,190	WOOFF ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 August 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *10,12,14,16-21,23,25,27-32,34,36,38-44,46 and 48-56* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, 48-56* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/6/08.

As indicated in Applicant's response, claims 10, 21, 32, 36, 43, 46, 53 have been amended. Claims 10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, 48-56 are pending in the office action, with the rest of the claims canceled.

Information Disclosure Statement

2. The information disclosure statement (IDS) letter submitted on 8/25/08 was filed and contains material (i.e. U.S. Application No. 10,728,370) related to the current Application. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, 48-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mathur, USPN: 5,008,814 (hereinafter Mathur), in view of APA (Admitted Prior Art: Background of Invention) and Vishwanath, USPubN: 2005/0198629 (hereinafter Vishwanath).

As per claim 10, Mathur discloses a method of software loading and initialization in a distributed network of nodes, the method comprising:

persistently storing, in a first storage of a master node, a plurality of software packages and a plurality of boot programs (e.g. Fig. 3; N_S, N_o – col. 5, line 3-20; *ILP* – col 7, lines 40-64 – Note: non-volatile storage for keeping cutover software for ILP in case of need reads on storage of boot package for regressing – see col. 3 line 65 to col. 4, line 49), wherein the plurality of software packages and the plurality of boot programs will be used by the nodes in the distributed network;

receiving, at the master node, a request for a boot program and software packages from a node, (e.g. *message receiver* – Fig. 3; request – col. 9 lines 9-53) in the distributed network, that is performing an initial boot;

the master node extracting the boot program and the one or more software packages from the first storage and delivering, to the node, the boot program and the one or more software packages (e.g. col. 5, line 31 to col. 6, line 54; step *202* - Fig. 2 – Note: delivering boot program and software packages reads on master node retrieving);

wherein said node:

stores the boot program and the one or more software packages in its local persistent storage (*initial boot program* – col. 3, line 32-41);

extracts software version information from the one or more software packages and stores the software version information in the local persistent storage (e.g. col. 3 line 65 to col. 4, line 49; *version ...release number* - col. 9, lines 45-52 – Note: each node storing of a trial version at local NVRAM reads on reboots using ILP software being downloaded from the N_S master node – see Fig. 3-- with storing extracted version thereof at the local node storage – N_d ... *collect*

information ... software versions - col. 8 lines 38-42 – prior to providing it back to the master task node);

reboots and executes the boot program stored in the local persistent storage (e.g. col. 7, lines 40-44), and

verifies the software version information with said master node (e.g. *send this information back to the operator ... software versions ... used by the master task* - col. 8 lines 34-56).

Mathur does not explicitly disclose persistently storing, in a second storage of the master node, software version information and node type information for each node in the distributed network; nor does Mathur explicitly disclose based on the request, the master node determining software version information of the node to retrieve from the second storage; the master node retrieving the software version information of the node from the second storage; and based on the software version information of the node, the master node determining, a boot image of the plurality of boot images and one or more software packages of the plurality of software packages to extract from the first storage. However, in order to provide the proper version of boot package to the requesting node, Mathur discloses server storing of information about the network node (e.g. col. 3, lines 21-32) purported for a update paradigm by which working versions from tried versions are selectively downloaded (see Summary; *version ...release number* - col. 9, lines 45-52), and in the course of which appropriate communication status and topological information are checked to ascertain that the target node environment is appropriately set for any boot program to be transmitted and operable therein (see Fig. 2; col. 5, line 3-30; col. 9, line 36 to col. 10, line 8); hence Mathur's emphasis on having proper information related to provisioning

working versions of boot programs to a requesting node is strongly implied. APA (Specifications: versions of Windows - pg. 1, bottom to pg. 2 top) teaches server's storing of versions made available for upgrade which can be periodically implemented to a multi-computer or multi-node system. At the time the invention was made, one of ordinary skill in the art, based on the purpose of Mathur's multi-node upgrade, would be motivated to enable storing at the master side sufficient node topological information, versioned programs (software/boot packages) with pertinent information as taught in APA for timely deriving which versions of programs should be provided to a particular node given the node definition in Mathur's knowledge about node topology respective to the node's request, for the benefits as set forth above, as to ensure proper component for a destined target, considering periodic need for upgrades as well-known in a multi-computer or multi-node topology as taught in APA.

Nor does Mathur explicitly disclose that the boot program being stored in the first storage is a boot image; and extracting of said boot image by the master node for delivery to the requesting node. But based on the message from a node to request a package requiring a ILP by Mathur, the need to extract boot program from a package being received for such request with which to effect an attempt to boot and effectuate basic O.S. initialization or activation (see: *fails to IPL, trial use* -col. 7, lines 15-63) is strongly implied. APA (Specifications: bottom pg. 2) teaches provisioning of boot program and Vishwanath, in a similar provisioning as APA and Mathur's ILP, teaches provisioning to a heterogeneous multi-node network and transferring of proper boot image (boot server – Fig. 3; step 407, Fig. 4; Fig 6) in conjunction with node request and rule-based extracting of the proper operating system program for booting. Based on Vishwanath(or APA) teaching of image for enabling client node to boot their system, it would

have been obvious for one skill in the art at the time the invention was made to implement the boot programs as boot image because as purported by Mathur's method to provisioning node as per a request basis of selective versions of working components would be facilitated by just sending an O.S. pertinent programs encompassed as an image (as by Vishwanath); thereby would obviate extraneous storage of a complete set of operating system components intended just for *initial program load* (IPL) at the target node; and this is the very intent by Vishwanath to use mapping rules in conjunction with server's database of booting components (see Vishwanath: para 0005 to 0008, pg. 1).

Mathur does not explicitly disclose: if said node does not have the correct software versions, then the master node retrieving correct software packages from the first storage and sending the correct software packages to said node; said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage.

However, Mathur discloses verification if correct version is stored prior to distribution then consistency is checked after it has been distributed (e.g. *a check is made ... whether ... with a new software version* - col. 5 lines 31 to col. 6 line 10), maintaining dialogue between a node trying to ILP and a operator node using a master node intermediary (Fig. 5-6; col. 7 line 40 to col. 8 line 22; col. 8 lines 34-56) whereby to follow progress of update, as to for example, validating on correctness of version being tried at the target node can be made via acknowledgement received from target node, wherein this continual acknowledging scheme would also assure retransmission of data based on acknowledgement of proper reception at the receiving node (col. 9 lines 12-20). For one of ordinary skill in the art, updating as in Mathur,

entails provisioning of the latest version to replace an older version. In a multi-node NW APA teaches provisioning of versions periodically, and Vishwanath's method teaches provisioning of correct components based on node/HW information (see TABLE 3, pg. 17-20) for enabling the node of a NW to boot into a target operating system (Fig. 4-8) and using a messaging protocol similar to Mathur, for the server to progressively track the node in terms as to whether a software package is appropriate with the target booting process, i.e. choosing the proper components to distribute to target computers (Vishwanath: Fig. 4-6, 12-14) based on dynamic validation. That is, similar to Mathur's open communication between a node and a master node or administrator, Vishwanath discloses administrator matching of validation rules in order to retrieve specific program (Fig. 12-14) or boot image (Fig. 4-6) and a communication scheme where the target node and the administrator maintain communication until all the requirements are satisfied (para 0154-0155, pg. 16); i.e. administrator acknowledging node's satisfaction in order to retrieve proper components via additional service provisioning or via incremental upgrade stages (see Fig 17 – Note: incremental provisioning **reads on** retrieving correct versions if acknowledge that previous sent version is not satisfactory). Based on upgrade of versions as set forth in Mathur and APA, such that proper components supporting completion of a node booting into a working state at a target node, and Mathur's maintained checking using the master node and operator regarding correctness of downloaded version in order for it to be permanently stored for activation at the target node (steps 202 – 204 Fig. 2), it would have been obvious for one skill in the art at the time the invention was made to implement Mathur's communication between the target node and the Master node (using operator directives) so that acknowledging of proper versioned package be implemented until all proper versioned upgrade components have been

incrementally provisioned or retransmitted using Vishwanath's approach as to fulfill the provision of upgrade components until all the required components for Mathur's node to ILP into a proper O.S. state; that is, when the node is not satisfied with the received components (as taught in Vishwanath – para 0155, pg. 16), the additional ILP components or more correct version being retrieved by the server or by invoking other provisioning services, so that more correct components be retransmitted to the target node, in order for Mathur's target node to complete the ILP intended via the above dialogue and acknowledgement approach, whereby a trial version can be finalized as complete and successful, based on the incremental provisioning by Vishwanath and the maintained update status communication approach by Mathur.

As per claim 12, Mathur discloses wherein said node, based on a command from said master node, does not store the one or more software packages in the local persistent storage device, allowing said master node to download test software packages (e.g. Fig. 2 – Note: master node transmitted version to local node reads on local node not having it stored locally prior to transmission – see step 204, step 211 – Fig. 2) to said node and temporarily run said node using the test software packages, and wherein when said node reboots, the test software packages will no longer exist on said node (step 210 – Fig. 2 – Note: trial run leading to a cutback reads on not having the bad trial software upon reboot – see col. 12, line 46-60 – where only one trial version remains after a failure and cutback).

As per claim 14, Mathur discloses wherein if said node has the correct software versions, then said node completes booting by executing one or more software packages stored in the local persistent storage (step 211, Fig. 2; col. 3 line 65 to col. 4, line 49).

As per claim 16, Mathur discloses wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration (e.g. hierarchical structure, privileges – col. 5, lines 3-27 – Note: structure representing grouping of nodes according to some particular privilege configuration reads on class of nodes of same configuration but different processors; see col. 3, lines 29-53) and may have differing processor types.

As per claim 17, Mathur does not disclose wherein each of one or more software package contains version information, dependency information, and other metadata information pertaining to software in the package; however teaches about administrator action based on topology of common path to node for routing, and hierarchy of privileges of nodes and checksum of software destined for distribution; as well as version and release of downloaded component (e.g. col. 9, lines 45-52; col 3, line 15-32; col. 5, lines 3-27; consistency check-- Fig. 2); hence the dependency over the network topological path, the access privileges and new version software information needed would be suggestive of metadata being collected via an administrative or server task for provisioning the above knowledge to informatively support the node distribution. Based on the boot image and storage of target system version and environment information by Vishwanath (refer to claim 1) wherein assembling a boot image implicates analysis, extraction, validation and retrieval of dependency specifics needed for packing components for boot deployment within a particular node OS environment (see Vishwanath: Fig. 3-10), it would have been obvious for one skill in the art at the time the invention was made to provide the administrator or server/source node– master node – with stored metadata relating to version information, dependency information as mentioned above so

that this information be packed in a target image as taught by Vishwanath; because this metadata would support the consistency checking as endeavored by Mathur in view of the topology-based for optimizing routing resources, expediting of the desired version of upgrade, and also for identifying access privilege per nodes as mentioned above.

As per claim 18, Mathur (in view of Vishwanath) discloses wherein a boot image is customized for a particular type of node and provides basic low-level communications (*initial boot program* – col. 3, line 32-41; Fig. 2 – Note: checking of checksum by master node in view of software to boot the node reads on boot image having node identification and low-level instructions, because without node type specificity is provided no proper reboot would be possible)

As per claim 19, Mathur (in view of Vishwanath) discloses boot images, software packages, and node information; and placing the boot programs and software packages in the first storage and the node information in the second storage on said master node (refer to claim 10); but Mathur does not explicitly disclose executing a composite image that is installed by a user onto said master node to create boot programs and packages and information; nor does Mathur disclose creating a subset of boot images, subset of plurality of software packages and placing these subset in first and second storage. The creation of boot program and node information being packaged for being extracted is disclosed in Mathur (re claim 10); and the administrative action to maintain version information and package for NW nodes update using operator's distribution command is taught (see *topology information*, *maintained* - col. 3, lines 29-53; col. 4, lined 64 to col. 5, line 27). Based on the creation of an image in Vishwanath's wherein package contents to support node booting are created via executing a rule-based

validation and target analysis to generate a image (see Vishwanath: Fig. 3-10), the limitation of user's installing at the master node and executing a composite program (or rule script) from an incremental obtaining of subsets of package requirement (Vishwanath – see para 0126 to para 0133, pg. 11-14) and image to create the final image as taught in Mathur would have been obvious; that is, one of ordinary skill in the art would be motivated to provide a administrating tool by Mathur and using Vishwanath's teachings to support necessary files extracted from a persistent storage of image subsets (Vishwanath: Fileserver 307 – Fig. 3; Fig. 8, 10; para 0039-0040, pg. 3-4; TABLE 1, pg. 4) to support a incremental gathering of a programmatic content inside each boot image (see Vishwanath: Fig. 4, 6, 12), provide execution by a composite process (see Fig. 14-15) by which the necessary boot programs, packages and type information (e.g. persisted in Mathur's topology information) stored at the master node to create the targeted node-specific boot image as set forth in the rationale of claim 10.

As per claim 20, Mathur (in view of Vishwanath) discloses wherein retrieving software version information creates the preferred software version information from the second storage based on functional features specified in the request by said node (see query handler - Fig. 4; *new version, identifies node ...privileges* - col. 5, line 3-30; col. 9, line 36 to col. 10, line 8 – Note: retrieving of proper version of boot program reads on storing of version based on stored privilege and/or identification information of nodes).

As per claim 21, Mathur (in view of Vishwanath) discloses a computer-readable medium carrying one or more sequences of instructions for software loading and initialization in a distributed network of nodes, which instructions, when executed by one or more processors,

cause the one or more processors to carry out the steps recited in claim 10; hence the rejection for such steps will incorporate the corresponding rejection as set forth therein respectively.

As per claims 23, 25, 27, refer to claims 12, 14, 16, respectively.

As per claims 28-31, refer to claim 17, 18, 19 and 20, respectively.

As per claim 32, Mathur (in view of Vishwanath) discloses an apparatus of software loading and initialization in a distributed network of nodes, comprising a master node and storage means with node information for supporting network node request and delivery including boot image and software packages as recited in claim 10; hence the rejection for all such limitations or means will incorporate the corresponding rejection as set forth therein respectively.

As per claims 34, 36, 38-42, refer to claims 23, 25, 27-31, respectively.

As per claim 43, Mathur discloses a system for software loading and initialization in a distributed network of nodes, the system comprising:

a master node;

a node in the distributed network;

a first storage on the master node, wherein the first storage persistently stores a plurality of boot programs and a plurality of software packages that nodes in the distributed network will use;

one or more processors on the master node (e.g. Fig. 1);

one or more sequences of instructions which, when executed by the one or more processors, cause the one or more processors to perform:

receiving a request for a boot program and software packages from the node that is performing an initial boot;

the master node determining, based on the request, a boot program of the plurality of boot program and one or more software packages of the plurality of software packages to extract from the first storage;

the master node extracting the boot program and the one or more software packages from the first storage; delivering, to the node, the boot program and the one or more software packages;

retrieving correct software packages from the first storage and sending the correct software packages to the node if the node does not have the correct software versions one or more other processors on the node (refer to the above extracting and delivering – Note: server sending extracted boot programs reads on correct packages or programs from the pertinent storage)

one or more other sequences of instructions which, when executed by the one or more other processors, cause the one or more other processors to perform:

storing the boot program and the one or more software packages in local persistent storage of the node;

extracting software version information from the software packages; storing the software version information in the local persistent storage; executing the boot program, that is stored in the local persistent storage, to reboot the node; verifying the software version information with the master node;

all of which having been addressed in claim 10.

Mathur does not explicitly disclose a second storage on the master node, wherein the second storage persistently stores software version information and node type information; and

based on the request, the master node determining and retrieving software version information of the node and node type information for each node in the distributed network from the second storage; nor does Mathur explicitly disclose that the boot program is a boot image and based on said software version information, extracting a boot image from the first storage based on said version information and delivering such boot image from the master node to network node.

But these limitations have been addressed in claim 10.

Nor does Mathur disclose completing a booting by the node, such as:

in response to receiving the correct software packages from the master node, storing the correct software packages in the local persistent storage and executing the correct software packages stored in the local persistent storage to complete booting.

But this limitation has been addressed in claim 10 above using Vishwanath and Mathur.

As per claim 44, refer to claim 12.

As per claim 46, refer to the rationale set forth in claim 14 regarding complete booting subsequent to receiving correct versions.

As per claims 48-52, refer to claims 18-20, respectively.

As per claim 53, Mathur does not explicitly disclose wherein:

the request includes node type information of the node;

the master node determining software version information of the node includes the master node using the node type information of the node to determine the software version information of the node.

But by virtue of the obviousness rationale in claim 1, based on storage in Mathur about node topology knowledge (Mathur: col. 3 lines 20-42) in combination with Vishwanath whereby

the master node uses the node type or hardware information to determine the software version or components destined for operating the node; e.g. discovery capability of the provisioning server by Vishwanath (see para00 37-0040; Fig 6; para 0126 to para 0133, pg. 11-14) to complement the version software for upgrade in node topology and version checking by Mathur as set in claim 1, would have motivated one of ordinary skill in the art to implement Mathur's server/master storage of NW topology so that this storage contains node information based on which to derive the proper and required boot components (e.g. proper version information) as discovered in Vishwanath's server storage system (see Vishwanath: Fig. 4).

As per claims 54-56, refer to the obviousness rationale of claim 53.

Response to Arguments

5. Applicant's arguments filed 8/6/08 have been fully considered but they are mostly moot and/or not persuasive. Following are the Examiner's observation in regard thereto.

USC § 103 Rejection:

(A) Applicants have submitted that step b) and d) being the newly added limitations to the claims are deemed not taught or rendered obvious by Mathur or Vishwanath (Appl. Rmrks pg. 18-20). The amendments have necessitated new grounds of rejection and the arguments are largely moot.

(B) Applicants have submitted that 'consistency check' as proffered by the Examiner is incorrect because the check is subsequent to receiving correct software and not related to version validation (Appl. Rmrks pg. 20). The limitations to address have elicited a new rationale as set forth in the Office Action.

(C) Applicants have submitted that nowhere in the cited parts of Mathur discloses or suggests extracting version information from packages being delivered from the master node (Appl. Rmrks pg. 21, middle pg. 22). The cited portions in Mathur discloses how a target node extract package for local storage and collect version information to have sent back to a master node in order to await further directives from an administrator; hence version information limitation being extracted and recollected has been met.

(D) Applicants have characterized interpretation by the Office Action as UNREASONABLE with regard to packet reception being proffered in the Office Action (Appl. Rmrks pg. 22). The allegation does not address a particular language of the claim and amounts to accusation not accompanied with facts as to clearly show how one particular language has not been met (or rendered obvious) by the Office Action, and that, taking into consideration that any particular rationale of rejection being questioned has to correspond (emphasis added) to a very claim limitation whose merits are addressed within a corresponding prosecution context or Office Action of record. That is, Claim 10 as amended and argued herein above encompasses new limitations; and the above allegations are not deemed commensurate with the grounds of rejection that addresses a former version of claim 10. The argument is deemed largely misplaced.

(E) Applicants have submitted that 'categorizing node into classes' was allegedly met with the *privilege access* in Mathur; and about the 'grouping' limitation as required in claims 16, 27, 38, 48 (Appl. Rmrks, pg. 24) the Office Action is incorrect. The Office Action has interpreted grouping in classes or categories in the way that some privileges can be grouped together in order to allow only a same privilege level or group of nodes (i.e. commonly having such level) to

be disseminated with the upgrades by Mathur. It is the burden of the Applicants to provide convincing facts as to show how the interpretation would be inappropriate, and how the language of the limitation has to be construed otherwise. Lacking proper or sufficient specificity (in the claim language) in terms of what grouping, categorizing, or class construction would be all about, it is deemed that the interpretation as set forth above is reasonable; and the argument is deemed non-persuasive. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

In all, the arguments are moot in view of the Amendements, or non-convincing because of the observations made above. The claims will stand rejected as set forth herein with the necessitated new Grounds of Rejection.

Interview Summary

6. The representative (Jason Lohr) and the Examiner have exchanged (8/21/08) discussions by phone and Emails regarding how to elaborate certain features of the independent claim with respect to a particularity in communication between the node and the master node in order to identify an improved scenario by which the case could be allowable. But no agreement has been reached.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

August 28, 2008